The Effect of Using Balancing Techniques on Phonemes Dataset

EE 660 Course Project

Project Type: (1) Design a system based on real-world data

Number of student authors: 1

Shoumik Nandi

shoumikn@usc.edu

12/06/2022

1. Abstract

In this work, I study the effect of algorithms that would help balance the data in the dataset and aid in distinguishing between nasal and oral sounds. I pose this as a binary classification problem. At first, I experiment by using the original dataset, which has approximately 71% oral and 29% nasal sounds. I trained machine learning classifiers to find the best model for the original dataset. I then implement various balancing algorithms: under sampling, oversampling, and combination sampling to obtain better results for the best model. Random Forest classifier with 1000 trees max features sqrt and gini criterion gave the best results with a test Geometric mean score of **0.911** and test accuracy of 0.913, with relative improvement of **27.59%** in geometric mean score and **19.03%** respectively, over a logistic regression baseline classifier. The corresponding Random Forest model with ADASYN oversampling balancing gave a relative improvement of **22%** in classification accuracy and **17.34%** in geometric mean score, over the baseline.

2. Introduction

2.1. Problem Type, Statement and Goals

The aim of this project is to distinguish between nasal (class 0) and oral (class 1) sounds. Five attributes were chosen to characterize each vowel: the amplitudes of the first five harmonics normalized by the total energy. The phonemes are transcribed as follows: **sh** as in she, **dcl** as in dark, **iy** as the vowel in she, **aa** as the vowel in dark, and **ao** as the first vowel in water.

The dataset is imbalanced so I will be evaluating different oversampling, undersampling and combination sampling algorithms with default hyperparameters. This should help with the balancing of the dataset by adding new examples for the minority class. I would largely be using Supervised learning algorithms and Semi-supervised learning algorithms for binary classification.

2.2. Our Prior and Related Work

None

2.3. Overview of Our Approach

In this project I will use supervised learning models and algorithms such as Dummy classifier with majority frequency strategy (majority class), Logistic Regression, Support Vector Machine with linear kernel, Decision Tree Classifier, and Random Forest Classifier. The results will be compared with metrics such as accuracy and Geometric mean score. Class labels will be predicted and both classes have equal importance. Hence I intend on using geometric means which is a metric that quantifies the performance of the model on both classes separately. I intend on selecting the best model based on the original dataset post which I wish to use balancing techniques to balance the original dataset using oversampling, undersampling, and combined sampling techniques, comparing results with the results prior to the balancing. I further intend on exploring semi supervised learning algorithms such as Label Propagation, Self Learning and Semi Supervised Support Vector machine models and compare their results using accuracy and geometric mean score.

2.3.1 Main topic (such as SL)

The baseline systems are

- The trivial baseline: Dummy classifier with majority frequency strategy (majority class)

- The non-trivial baseline: Logistic regression

2.3.2 Extension (SSL)

The baseline system: Self Training using Logistic Regression

3. Implementation

3.1. Data Set

The dataset used is the Phonemes Dataset from KEEL dataset repository. https://sci2s.ugr.es/keel/dataset_smja.php?cod=1489 : Link to obtain the dataset. This dataset contains five phonemes (Aa, Ao, Dcl, Iy, Sh) with numerical values as attributes. The attributes are all of type float. There are a total of 5404 data points. The aim of the dataset is to distinguish between nasal (class 0) and oral sounds (class 1). Hence it is a binary classification.

Attribute	Domain
Aa	[-1.7, 4.107]
Ao	[-1.327, 4.378]
Dcl	[-1.823, 3.199]
Iy	[-1.581, 2.826]
Sh	[-1.284, 2.719]
Class	{0, 1}

Table 1: Dataset Attribute Description

There are three files for the dataset:

'phoneme-ssl40-10-1tra.dat': This file contains data points that are used for training the semi-supervised learning algorithms. This 40% of the data points are labeled whereas the remaining 60% is unlabeled. This contains 90% of the total data points (4863) in the dataset.

'phoneme-ssl40-10-1trs.dat': This file contains data points that are used for training supervised learning algorithms. All the data points are labeled. This contains 90% of the total data points(4863) in the dataset.

'phoneme-ssl40-10-1tst.dat': This file contains data points that are used for testing both the semi-supervised and the supervised learning algorithms. All the data points contain their label. This contains 10% of the total data points (541) in the dataset.

3.2. Dataset Methodology

The original dataset is divided into training and testing datasets in which 90% of the data is training data and 10% data is testing data. I will be using repeated stratified k-fold cross-validation to evaluate the models. I will use k=10, meaning each fold will contain 4863/10 i.e. 486 samples. Using stratified k-fold cross validation will help in dividing each fold into the original proportion of the training dataset i.e. approximately 71% nasal and 29% oral sounds. I will use five repeats. Repetition indicates that the process of evaluation will be performed five times over to avoid fluke results and capture the variance of the chosen model. The test set will be used to evaluate the performance of the model on unseen data.I will use both the metrics accuracy and geometric mean score to see which model gives us the best results.

3.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

The dataset does not have any missing values. The original dataset contains 71% nasal (class 0) sounds and 29% oral (class 1) sounds. Both class labels are of equal importance. As a result, I selected a metric which quantifies the performance of both classes separately. G mean score can be calculated:

G-mean = sqrt(Sensitivity * Specificity)

where sensitivity measures accuracy of positive class and specificity measures the accuracy of the negative class

Specificity = TN / (TN + FP)

Sensitivity = TP/ (TN+FP)

where TN = True Negative , TP = True Positive, FP = False Positive

3.3.1 Exploratory Data Analysis

In this subsection, I analyze and present the data analysis techniques I used on the original dataset. Exploratory data analysis helps us in analyzing trends and patterns that are present in the training data. This way, the required preprocessing steps are taken.

Histogram

I used histogram to look at the distribution of the five numerical input variables in the dataset along with the class label. The histograms for each of the five numerical variables are plotted classwise. We can see that the variables appear to have a Gaussian distribution. This can be observed in the figure below.

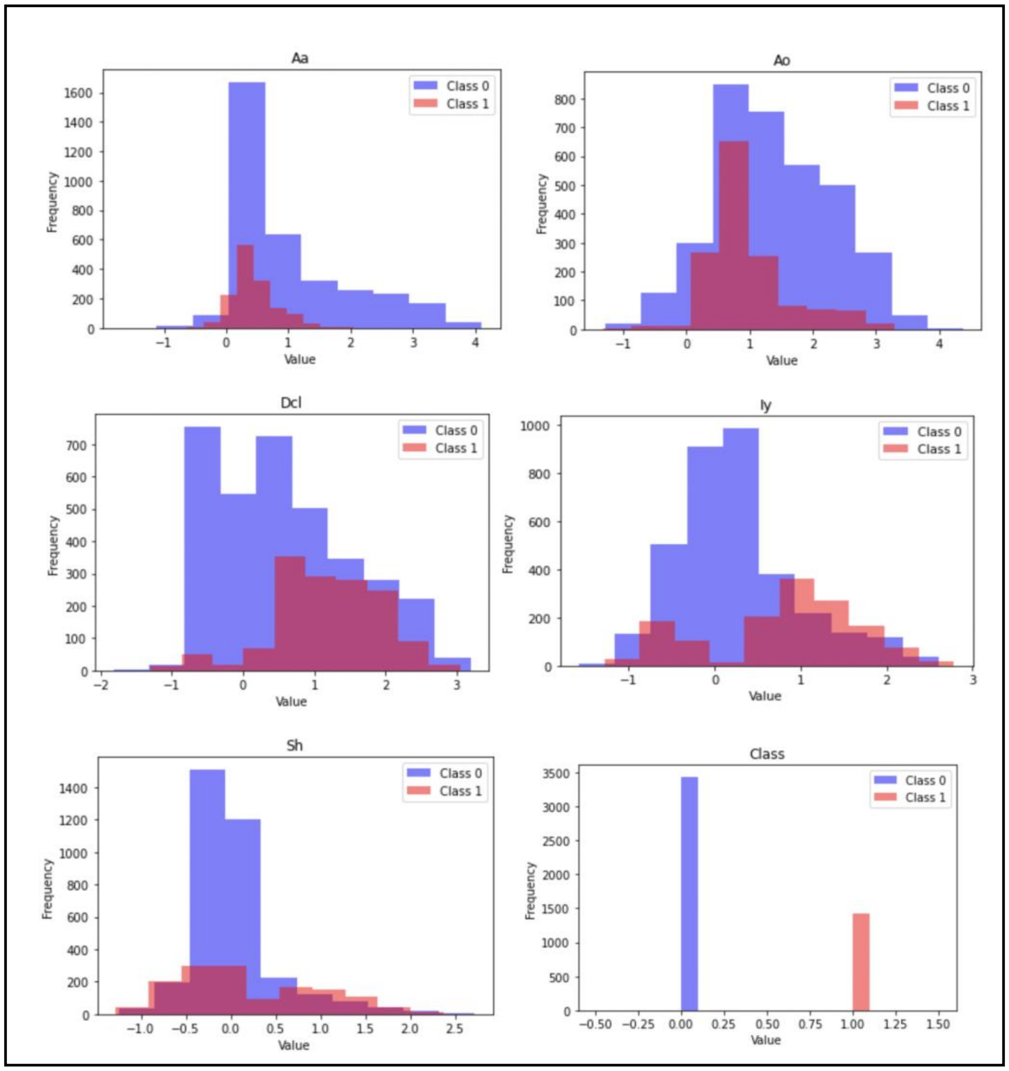


Figure 1: Histogram of features(class-wise)

Scatter Plot Matrix

I used a scatter plot matrix to compare each of the five input variables (attributes) with each other. The diagonal of the matrix shows the density distribution of each variable. I observed that the distribution for the variables differ with classes. As a result I can possibly discriminate based on classes. This can be observed in the figure below.

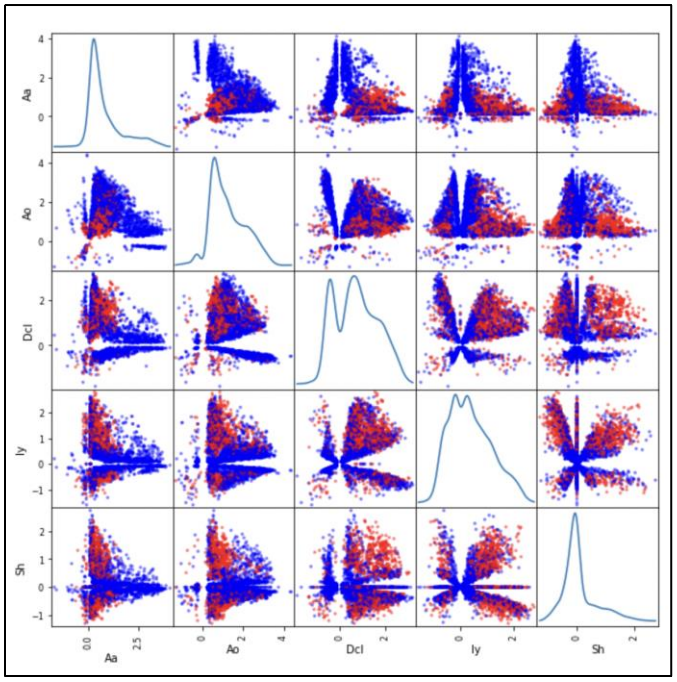
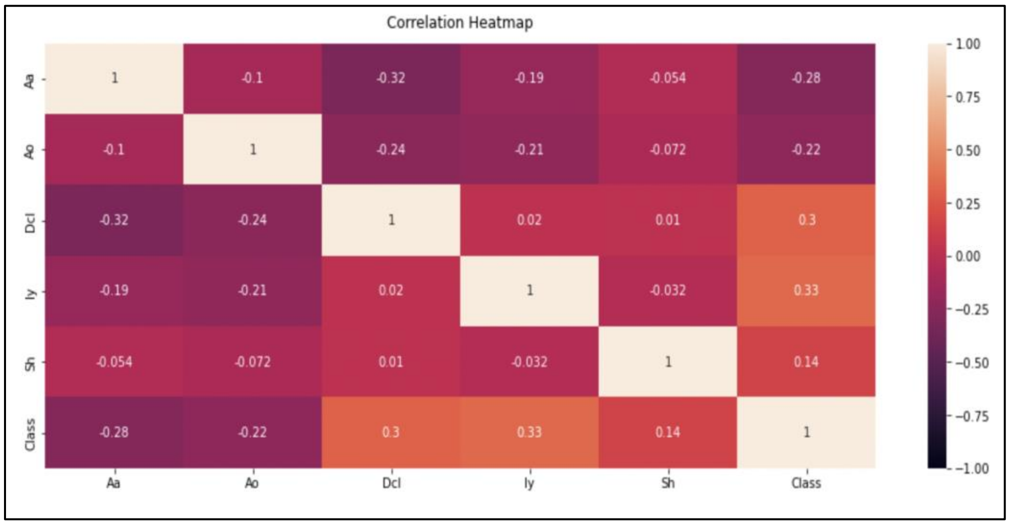


Figure 2: Feature Scatter Matrix

Heatmap

I used heatmap to visualize correlation between input variables as well as the class. On observation I could conclude that none of the variables are heavily correlated due to which I decided to keep all the features and reduce the number of features. This can be observed in the figure below.



Correlation Heatmap

	Aa	Ao	Dcl	Iy	Sh	Class
Aa	1	-0.1	-0.32	-0.19	-0.054	-0.28
Ao	-0.1	1	-0.24	-0.21	-0.072	-0.22
Dcl	-0.32	-0.24	1	0.02	0.01	0.3
Iy	-0.19	-0.21	0.02	1	-0.032	0.33
Sh	-0.054	-0.072	0.01	-0.032	1	0.14
Class	-0.28	-0.22	0.3	0.33	0.14	1

Figure 3: Covariance heatmap of the original train set

3.3.2 Standardization

I decided to carry out standardization since there is a difference in range of the input variables. While using models that are based on distance computation having different ranges can cause a bias and the feature with the greater range can possibly govern the distance.

Feature standardization is given by the formula:

$$x' - \frac{x - \mu}{\sigma}$$

3.3.3 Balancing Techniques

The original dataset contains 71% nasal (class 0) sounds and 29% oral (class 1) sounds. The dataset seems to be unbalanced. I decided to apply a few balancing algorithms with the aim of analyzing its effect on binary classification.

Oversampling Techniques

- Random Oversampling

  Random oversampling involves randomly selecting examples from the minority class, with replacement, and adding them to the training dataset.

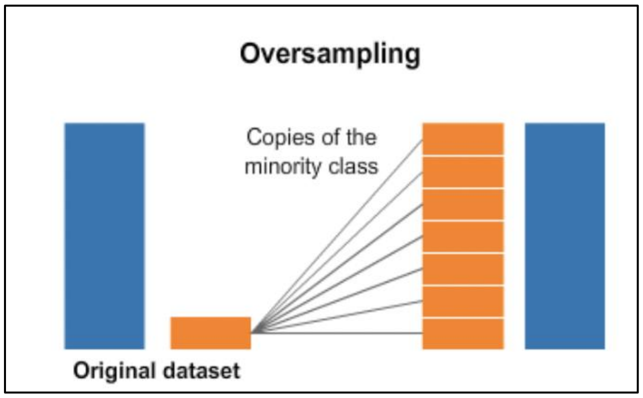


Figure 4: Random Oversampling

- SMOTE

  Synthetic Minority Over Sampling Technique (SMOTE) algorithm applies KNN approach where it selects K nearest neighbors, joins them and creates the synthetic samples in the space. The algorithm takes the feature vectors and its nearest neighbors, and computes the distance between these vectors. The difference is multiplied by a random number between (0, 1) and it is added back to the feature.

I used the default parameters which use k = 5.

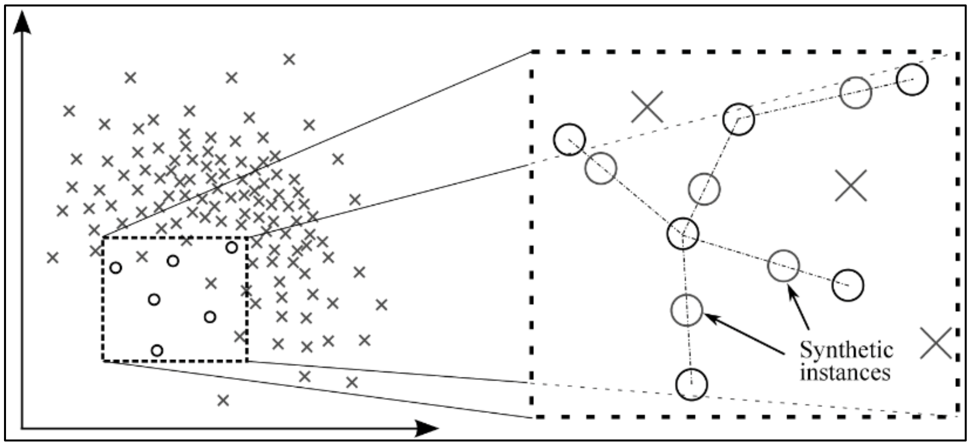


Figure 5: SMOTE

- ADASYN

The algorithm adaptively updates the distribution and there are no assumptions made for the underlying distribution of the data. The algorithm uses Euclidean distance for KNN Algorithm. The key difference between ADASYN and SMOTE is that the former uses a density distribution, as a criterion to automatically decide the number of synthetic samples that must be generated for each minority sample by adaptively changing the weights of the different minority samples to compensate for the skewed distributions.

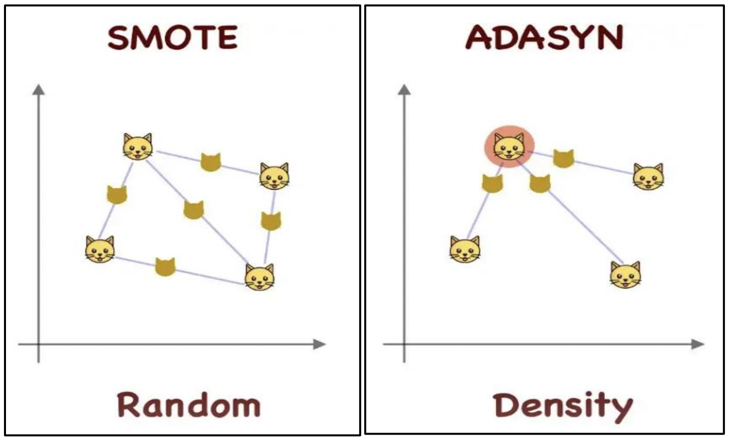


Figure 6: SMOTE vs ADASYN

Undersampling Techniques

- Random Undersampling

  Random undersampling involves randomly selecting examples from the majority class and deleting them from the training dataset.

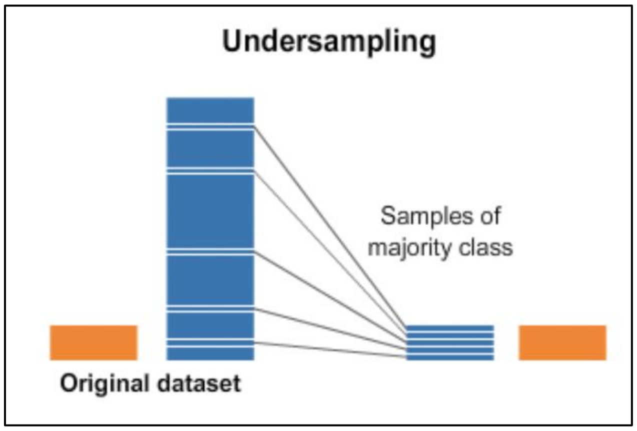


Figure 7: Random Undersampling

- Tomek Links

  Tomek Links are pairs of examples of opposite classes in close vicinity. This method ends up removing majority class data points from the Tomek Link, which provides a better decision boundary for a classifier there by removing the ambiguous data points.

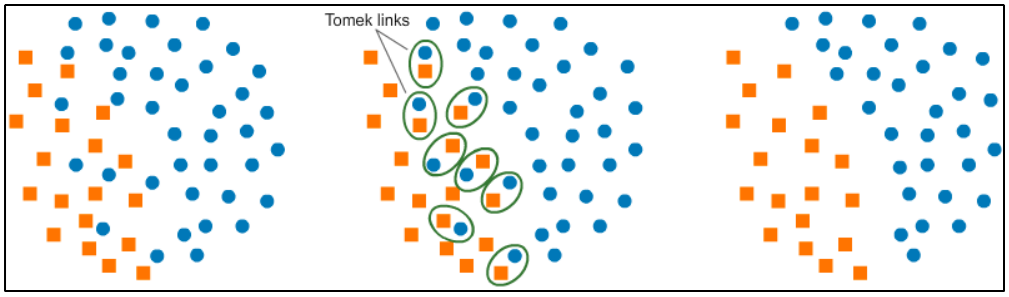


Figure 8: Tomek Links

- Edited Nearest Neighbors

  Edited Nearest Neighbors Rule for undersampling, K=3 nearest neighbors is used to identify data points that are misclassified. These data points are then removed before a K=1 classification rule is applied. This rule can be applied to each example in the majority class, allowing those examples that are misclassified as belonging to the minority class to be removed and those correctly classified to remain.

Combined Sampling Techniques

Data undersampling will delete examples from the majority class, whereas data oversampling will add examples to the minority class. These two approaches can be combined and used on a single training dataset. This will ensure that the data points are well differentiated thereby preventing overfitting problems.

- SMOTETomek

  Combine over- and under-sampling using SMOTE and Tomek links.

- SMOTEEN

  Combine over- and under-sampling using SMOTE and Edited Nearest Neighbors.

3.4. Training Process

3.4.1 ML classifiers used

- **Trivial Baseline (DummyClassifier(strategy:”uniforml”)) :** In this model, an ”unseen” point is randomly classified into one of the two classes with equal probability thereby giving a Geometric score of 0.5.

  In my work, I directly used sklearn’s dummy.DummyClassifiermodule and set strategy = “uniform”

- **Logistic Regression (Non-Trivial Baseline):** Logistic regression applies a sigmoid activation function using a threshold to estimate the probability of a binary class the datapoint belongs to.
- **Support Vector Machines (SVM)**: The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that I can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors. I was interested in understanding the effect of two kernels of SVM i.e., linear and radial basis function (RBF). With a linear kernel, a single straight line can classify the two classes. However, an RBF kernel makes use of a Gaussian similarity function, with an aim of modelling the similarity between a test point and the training point in a Gaussian function as,

$$K(x_1, x_2) = exp(\frac{\|x_1 - x_2\|^2}{2\sigma^2})$$

This model uses kernel transformations to find optimal boundaries and I expected it to yield better results than the baseline.

In my work, I directly used sklearn's svm.SVC and set kernel = linear and rbf.

- **Decision Tree**: Decision Tree is a supervised learning algorithm and can be used for both regression and classification. For my work, I use it for the purpose of classification. A DT creates a model that predicts the class of the target variable by learning decision rules with the help of the training data. The root node of the tree represents the entire set of data. This data gets divided into homogeneous sets. Nodes after which further splitting does not take place are called leaf nodes. The decision tree sorts data as it goes down the tree from the root node to the leaf node. The classification of the datapoint is given at the root node. Each node acts as a test case for some attribute and each edge descending to its sub nodes corresponds to the possible answers of the test. This process goes on recursively until a leaf node is reached and the datapoint is classified. To decide which feature to select for a particular node to perform a test on the data, there are attribute selection methods. The two famous criterion functions are entropy and gini impurity. I used the gini impurity in our work. Gini impurity is calculated based on the gini index which can be viewed as a cost function which is calculated by subtracting the square of probability of each class from one, and favors larger partitions.

$$gini = 1 - \sum_{i=1}^{C} P_i^2$$

Lower the value of the gini index, better the criteria for splitting and higher the homogeneity. The depth of the decision can be predefined if I want to. Based on this the tree will stop splitting the data and will give me the classification once the predefined depth is reached.

I used this model since it is expected to yield very good results.

I directly use sklearn's implementation of DecisionTreeClassifier and set the criterion function as 'gini'.

- **Random Forest:** Random forest is a supervised learning algorithm which is used for both regression and classification. In this work, I am using it for classification. Random forest builds decision trees using different data and takes a majority vote for classification. I can predefine the number of decision trees I want the RF model to use to carry out the classification. It uses an ensemble technique called bagging. Ensemble means combining multiple classifiers and using this collection to carry out prediction. Bagging chooses random data points from the dataset with replacement. Each tree is trained independently, and results are generated. The final classification is done based on majority voting after combining the results of all the classifiers. The RF model in my work uses the gini impurity for feature selection for all the decision trees in the model. Following are the steps:

  1. k random points are taken from all the training datasets.

  2. Individual decision trees are constructed from each of the k selected random points.

  3. Each decision tree will generate an output.

  4. Final class assigned based on majority voting

  I used this model since it is expected to yield better results than a single decision tree.

  In my work, I directly used sklearn's ensemble.RandomForestClassifier module and set n estimators = 1000 and the criterion function as 'gini' after finding these parameters using Grid Search.

3.4.2 Experimental Setup

- The Training dataset '**phoneme-ssl40-10-1trs.dat**' and Test dataset '**phoneme-ssl40-10-1tst.dat**' were used to observe whether dataset balancing techniques would improve model results. The data was first standardized and then machine learning models were trained and evaluated on the original data. Via these results, the best model would be optimized using GridSearchCV. After obtaining the optimal parameters for the selected model, balancing techniques : Undersampling , Oversampling and Combination Sampling would be utilized to balance the data and then run the optimized model. The following subsection reports the average validation accuracy and the geometric mean score per model, using stratified 10 fold cross validation. Sklearn's inbuilt functions were used to calculate the accuracy and the geometric mean score. Based on the cross validation geometric mean score the best model would be selected through which the testing results would be obtained.

3.5. Model Selection and Comparison of Results

The average validation accuracy and the average score for the original dataset.

Name of Model	Mean Validation Accuracy	Mean Validation Geometric Mean Score
Trivial classifier	0.501	0.500
Logistic Regression (Baseline)	0.748	0.767
SVM Linear Kernel	0.772	0.700
SVM rbf kernel	0.841	0.801
Decision Tree	0.867	0.838
Random Forest	**0.907**	**0.881**

Table 2: Training Results for Original Training set

Random Forest Classifier had the highest Geometric Mean Score on the original dataset. Hence I considered optimizing this model to get better results using GridSearchCV. Grid search was carried out using Geometric mean score as the scoring method and it was carried out across 3 parameters :

n_estimators: [100, 200, 500, 1000, 1500]

max_features: [‘sqrt’, ‘log2’]

criterion: [‘gini’, ’entropy’]

On completion on grid search I got the following result:

Name of Model	n_estimator	max_features	criterion	Mean Validation Accuracy	Mean Validation Geometric Mean Score
Random Forest	**1000**	**sqrt**	**gini**	**0.91**	**0.884**

Table 3: Training Results for Optimizing best model -> Random Forest

In an attempt to get a better result , I decided to try different balancing techniques and then evaluate the results using the Random Forest Classifier with the best parameters that I found using GridSearchCV using Geometric Mean Score as the metric.

Balancing Using Oversampling Techniques

Name of Model	Oversampling Technique	Mean Validation Accuracy	Mean Validation Geometric Mean Score
Random Forest	ROS	0.906	0.895
Random Forest	SMOTE	0.901	0.896
Random Forest	**ADASYN**	**0.893**	**0.896**

Table 4: Training Results for Training set after Balancing using Oversampling

Balancing Using Undersampling Techniques

Name of Model	Undersampling Technique	Mean Validation Accuracy	Mean Validation Geometric Mean Score
Random Forest	RUS	0.872	0.883
Random Forest	**Tomek Links**	**0.907**	**0.889**
Random Forest	Edited Nearest Neighbors	0.867	0.881

Table 5: Training Results for Training set after Balancing using Undersampling

Balancing Using Combined Sampling Techniques

Name of Model	Combined Technique	Mean Validation Accuracy	Mean Validation Geometric Mean Score
Random Forest	**SMOTETomek**	**0.901**	**0.894**
Random Forest	SMOTEEN	0.860	0.871

Table 6: Training Results for Training set after Balancing using Combined Sampling

My main objective was to train the model and calculate the mean validation accuracy and mean validation geometric mean score over the different balancing techniques and choosing the best one best on the geometric mean score. I expected that the sampling techniques would result in a lift in the performance. Via the results I could observe that **ADASYN** gives the highest geometric mean score for oversampling **(0.896)**, **Tomek Links** gives the highest geometric mean score for under sampling **(0.889)** and **SMOTETomek** gives the highest geometric mean score for combined sampling **(0.894)**. The highest overall geometric mean score is given by ADASYN. Hence chose Random Forest Classifier with ADASYN oversampling as the final model.

4. Final Results and Interpretation

Name of Model	Test Accuracy	Test Geometric Mean Score	Confusion Matrix
Trivial Classifier	0.4658	0.426	[[177 84] [205 75]]
Logistic Regression	0.767	0.714	[[335 79] [47 80]]
SVM linear kernel	0.7966	0.751	[[334 62] [48 97]]
SVM rbf kernel	0.874	0.850	[[352 38] [30 121]]
Decision Tree	0.832	0.794	[[338 47] [44 112]]
Random Forest	**0.913**	**0.911**	**[[369 34] [13 125]]**

Table 7: Results for Testing set for Original Dataset

Name of Model	n_estimators	max_features	criterion	Test Accuracy	Test Geometric Mean Score	Confusion Matrix
Random Forest	**1000**	**sqrt**	**gini**	**0.911**	**0.909**	**[[369 35] [13 124]]**

Table 8: Test Results for Optimizing best model -> Random Forest

Name of Model	Oversampling Technique	Test Accuracy	Test Geometric Mean Score	Confusion Matrix
Random Forest	ROS	0.902	0.888	[[361 32] [21 127]]
Random Forest	SMOTE	0.904	0.882	[[356 26] [26 133]]
Random Forest (Final Model)	**ADASYN**	**0.900**	**0.874**	**[[351 23] [31 136]]**

Table 9: Results for Testing set after Balancing using Oversampling

Name of Model	Oversampling Technique	Test Accuracy	Test Geometric Mean Score	Confusion Matrix
Random Forest	RUS	0.898	0.866	[[342 15] [40 144]]
Random Forest	**Tomek Links**	**0.9168**	**0.9066**	**[[365 28] [17 131]]**
Random Forest	Edited Nearest Neighbors	0.881	0.845	[[333 15] [49 144]]

Table 10: Results for Testing set after Balancing using Undersampling

Name of Model	Combined Technique	Test Accuracy	Test Geometric Mean Score	Confusion Matrix
Random Forest	**SMOTETomek**	**0.901**	**0.894**	**[[335 15] [47 144]]**
Random Forest	SMOTEEN	0.860	0.871	[[357 24] [25 135]]

Table 11: Results for Testing set after Balancing using Combined sampling

The baseline model had a mean cross validation accuracy of **0.748** and mean validation geometric mean score of **0.767** . The baseline model has a testing accuracy of **0.767** and a testing geometric mean score of **0.714**. From the training metrics we observed that ADASYN oversampling with the Random Forest Classifier gives the best training results. The training accuracy is **0.893** and a geometric mean score of **0.896**. The testing accuracy is **0.9** and the testing geometric mean score is **0.874**. These values are significantly higher than the baseline. The training geometric mean is the highest for ADASYN. The test geometric mean score is also pretty. ADASYN adaptively generates minority data based on their distributions against K nearest neighbors. The advantage of this is that it compensates for a skewed distribution. By comparison SMOTE generates a fixed ratio of synthetic samples for each minority data sample.

5. Implementation for the extension

Extension includes implementation of Semi-supervised learning algorithms

5.1. Data Set

Dataset has been discussed in section 3.1. Since I am implementing Semi-Supervised Learning I will be using the '**phoneme-ssl40-10-1tra.dat**' dataset for training. 40% of the data points are labeled whereas the remaining 60% is unlabeled. This contains 90% of the total data points(4863) in the dataset. I will be using the '**phoneme-ssl40-10-1tst.dat**' dataset for testing the models. This contains 10% of the total data points(541) in the dataset.

5.2. Dataset Methodology

The test set will be used to evaluate the performance of the model on unseen data. We will use both the metrics accuracy and geometric mean score to see which model gives us the best results. Reason for using geometric mean score in section 3.3

5.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

5.3.1 Standardization

We also carry out standardization before inputting the input variables to models. The description of it has been discussed in section 3.3.2.

5.4. Training Process

5.4.1 ML classifiers used (Semi-supervised Learning)

- **Self Training** : Self Training is a semi supervised learning model where we train a model based on the labeled data points using a supervised learning model (used Logistic Regression in this case). The training set contains labeled and unlabeled datapoints. In this case logistic regression is used to train the model using the labeled datapoints and then this model is used to predict the labels of the unlabeled datapoints in the training set. Each unlabeled datapoint has a predicted label which is predicted with a certain level of confidence. If the level of confidence/ probability is greater than a particular threshold, the datapoint along with the predicted label is added to the training dataset as a labeled datapoint. After this the new labeled training set is used for training. We iterate through this process again and again till no new points are added with their predicted label. Now this final set of labeled points are used to train the model and this model is used for predicting the output. There are unlabeled points which remain unlabeled since they never reach the confidence level. Here I have used the ad hoc condition where the unlabeled datapoints are given the label of the majority class.

- **Label Propagation**: Label Propagation is attempting to assign labels based on the set of known labels. In my work, I directly used sklearn's semi_supervised.LabelPropagation and set kernel = **knn.** The default nearest neighbors considered is 7. The algorithm is effectively making connections between all data points and giving the connection edge a weight based on level of connection. Next the algorithm iteratives through a **random walk** starting from each unlabeled datapoint. This process continues until all walk paths have been explored. Labels are assigned based on the length of the walk until reaching a labeled point. Length is inversely proportional to the weight.
- **Semi-Supervised Support Vector Machine:** The Semi supervised support vector machine accomplishes label propagation in a different way compared to **knn**. S3VM first builds a classifier based on the known labels. It then performs inference on the unlabeled data. Classifications with high confidence will assign their label to the data. The classifier is then retrained based on the new labels. This continues until all possible labels are assigned. Based on the confidence threshold it is possible to have unassigned labels on completion.

5.4.2 Experimental Setup

- The training data set '**phoneme-ssl40-10-1tra.dat**' and testing dataset '**phoneme-ssl40-10-1tst.dat**' will be used to see how semi supervied models will work on them. The dataset used has a split of 40% labeled to 60% unlabeled data in the training set. The dataset would be trained and then the results would be evaluated. The Self Training algorithm using Logistic Regression was the baseline model and then improvements would be tried on it by using other algorithms such as the Label Propagation and the S3VM. The following subsection reports the average train accuracy and the geometric mean score per model. sklearn's inbuilt functions were used to calculate the accuracy and the geometric mean score. The training accuracy and the training geometric mean score were calculated by comparing the results to the training set label used for supervised learning training '**phoneme-ssl40-10-1trs.dat**'

5.5. Model Selection

I used the Self training model with the Logistic Regression as the baseline model.

Model	Train Accuracy	Train Geometric Mean Score
Self Training - Logistic Regression (Baseline)	0.733	0.512
Label Propagation	**0.872**	**0.856**
S3VM	0.774	0.719

Table 12: Training Results using SSL algorithms

I used geometric mean score as the metric to select the best model. On training the three models I observed that label propagation gives the best results. The label propagation model uses the knn kernel and used k=7 which are the default parameter to train.

6. Results and Interpretation for the extension

Model	Test Accuracy	Test Geometric Mean Score	Confusion Matrix
Self Training - Logistic Regression (Baseline)	0.762	0.642	[[338 85] [44 74]]
Label Propagation	**0.863**	**0.848**	**[[358 50] [24 109]]**
S3VM	0.798	0.753	[[334 61] [48 98]]

Table 13: Test Results using SSL algorithms

On evaluating the model on the test set we can see that the best model given to us by the training set gives us the best result. It gives far higher results in terms of the metrics selected, which is the geometric mean score. In the case of S3VM it gives higher results than the baseline. I also tried training an SVM model on the labeled data of the training set i.e. training a supervised model on 40% of the training set. The S3VM model tends to do better after using both the labeled and unlabeled data compared to a model only using the labeled data in a supervised model case. We can see the advantage of using a semi supervised model in this case.

7. Summary and conclusions

Supervised Learning

- Random Forest Classifier gives the best performance compared to the other algorithms.
- When data is unbalanced we should use Stratified K fold Cross Validation since at each fold the mixture of data will be of the same proportion as the original training dataset.
- GridSearchCV is an efficient way to find the best parameters, given a list of parameters. In this case we are able to find the best n_estimators, the maximum features and the criterion to measure the quality of a split using geometric mean score as a metric.
- Balancing the dataset increases the performance of the model since it avoids the model from becoming biased towards one class.
- It is necessary to standardize data before balancing since many of the balancing algorithms use distance to carry out sampling.

Future Scope

- Experimentation can be done using other supervised learning models to get a better fit.
- More combined sampling methods can be tried to increase the performance of the model by pipelining an oversampling and an undersampling method.
- More features can be generated as a combination of two or more features to see if it helps improve the model.

Semi- Supervised Learning

- Label Propagation gives the best performance compared to other algorithms.
- Data must be standardized or normalized before using certain algorithms since they use k nearest neighbors algorithm to build on. Hence if features have different bounds one feature might affect the result way more than the other.
- Semi-supervised models are extremely useful in utilizing unlabeled data and do give equal or better results even after being unlabeled. It is necessary to choose the correct semi-supervised classification algorithms since they are highly

sensitive to data distribution. A wrong model can give a completely different classifier which can yield extremely poor results.

Future Scope

- Implementation of other algorithms such as Harmonic Function algorithm, Expectation maximization and other algorithms.

8. References

[1] J. Brownlee, "Undersampling algorithms for imbalanced classification," *MachineLearningMastery.com*, 26-Jan-2021. [Online]. Available: https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/

[2] Steen, D. (2020) *A gentle introduction to self-training and semi-supervised learning*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/a-gentle-introduction-to-self-training-and-semi-supervised-learning-ceee73178b38

[3] Brownlee, J. (2021) *Imbalanced classification with python (7-day mini-course)*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/imbalanced-classification-with-python-7-day-mini-course

[4] *Smotetomek#* (no date) *SMOTETomek - Version 0.9.1*. Available at: https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTETomek.html

[5] *The 5 most useful techniques to handle imbalanced datasets* (no date) *KDnuggets*. Available at: https://www.kdnuggets.com/2020/01/5-most-useful-techniques-handle-imbalanced-datasets.html

[6] Bhattacharyya, I. (2022) *Smote and ADASYN (handling imbalanced data set)*, *Medium*. Coinmonks. Available at: https://medium.com/coinmonks/smote-and-adasyn-handling-imbalanced-data-set-34f5223e167

[7] Brownlee, J. (2021) *Tour of evaluation metrics for imbalanced classification*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification .

[8] Brownlee, J. (2021) *Smote for imbalanced classification with python*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification